		OR			
9	a)	Describe various software quality models and the difference between product metrics and process metrics.	L2	CO1	5 M
	b)	Explain the role and importance of software documentation, distinguishing between internal and external documentation.	L2	CO1	5 M
		UNIT-V			
10	a)	Explain the characteristics of a CASE tools and their significance.	L2	CO1	5 M
	b)	Elaborate on the different software maintenance process models. How do these models help in estimating the cost and effort required for maintenance activities?	L4	CO4	5 M
		OR			
11	a)	Describe the architecture of a CASE environment. What are the key components and how do they interact to support software development and maintenance?	L2	CO1	5 M
	b)	Explain the concept of software reverse engineering in detail.	L2	CO1	5 M

PVP 23

Code: 23DS3503

III B.Tech - I Semester - Regular Examinations - NOVEMBER 2025

SOFTWARE ENGINEERING

(CSE - DS)

Duration: 3 hours Max. Marks: 70

Note: 1. This question paper contains two Parts A and B.
2. Part-A contains 10 short answer questions. Each Question carries 2 Marks.
3. Part-B contains 5 essay questions with an internal choice from each unit. Each Question carries 10 marks.
4. All parts of Question paper must be answered in one place.

BL – Blooms Level CO – Course Outcome

PART – A

		BL	CO
1.a)	Define Software Crisis.	L2	CO1
1.b)	List two notable changes in software development practices in recent years.	L2	CO1
1.c)	What is the requirement of Engineering Process?	L2	CO1
1.d)	What are the key responsibilities of a software project manager?	L2	CO1
1.e)	Define cohesion and coupling in software design.	L2	CO1
1.f)	State any two characteristics of a good user interface.	L2	CO1
1.g)	Define the difference between internal and external software documentation.	L2	CO1
1.h)	Differentiate between verification and validation in software testing.	L2	CO1
1.i)	What are two major benefits of using CASE tools in software development?	L2	CO1
1.j)	What is component classification in software reuse?	L2	CO1

PART – B

			BL	CO	Max. Marks
		UNIT-I			
2	a)	What is the use of software development process models? Explain.	L2	CO1	5 M
	b)	Compare and contrast the Iterative Waterfall Model and the Incremental Development Model.	L2	CO1	5 M
		OR			
3	a)	Describe the types of software development projects with suitable examples.	L2	CO1	5 M
	b)	Illustrate the working of the Rapid Application Development (RAD) model. What types of projects are most suited for RAD, and why?	L2	CO1	5 M
		UNIT-II			
4	a)	Write short note on Requirement Specification and Requirement Validation.	L2	CO1	5 M
	b)	Compare and contrast Lines of Code (LOC) and Function Point (FP) metrics for project size estimation. In what scenarios would you prefer one metric over the other?	L4	CO4	5 M
		OR			
5	a)	Requirements analysis is unquestionably the most communication intensive step in the software engineering process. Why the communication path does frequently breaks down?	L3	CO2	5 M
	b)	Explain the Basic COCOMO Model. How does it estimate effort and cost for software projects? Support your answer with examples.	L4	CO4	5 M
		UNIT-III			
6	a)	Compare and contrast the two primary approaches to software design: Function-Oriented Design and Object-Oriented Design. Discuss their key characteristics, advantages, and disadvantages	L4	CO4	5 M
	b)	Describe the Extreme Programming (XP) process. What are the key practices that differentiate XP from other agile methodologies?	L2	CO1	5 M
		OR			
7	a)	Explain the Scrum framework in detail. Describe the roles, artifacts, involved in a typical Scrum project.	L2	CO1	5 M
	b)	Discuss the characteristics of a Good User Interface (UI). Why is a well-designed UI critical for the success of a software application?	L3	CO3	5 M
		UNIT-IV			
8	a)	Describe black-box testing techniques, focusing on equivalence class partitioning and boundary value analysis.	L2	CO1	5 M
	b)	What is the ISO 9000 certification? How does ISO 9000 apply to the software industry, and how does it compare with the SEI Capability Maturity Model (CMM)?	L3	CO3	5 M

Code: 23DS3503

PVP 23

III B.Tech. – I Sem- Regular Examinations
NOVEMBER 2025

SOFTWARE ENGINEERING

CSE (Data Science)

1.(CO1, L2,2M)

1a) Software Crisis

Definition – 2M

1.b) Notable changes

Any two points – 1M + 1M

1.c) Requirement of Engineering Process

Purpose/need – 2M

1.d) Responsibilities of Project Manager

Any two responsibilities – 1M + 1M

1.e) Cohesion & Coupling

Cohesion definition – 1M

Coupling definition – 1M

1.f) Characteristics of good UI

Any two points – 1M + 1M

1.g) Internal vs External Documentation

Internal documentation – 1M

External documentation – 1M

1.h) Verification vs Validation

Verification – 1M

Validation – 1M

1.i) Benefits of CASE Tools

Any two benefits – 1M + 1M

1.j) Component Classification

Definition/meaning – 2M

2(a) What is the use of software development process models? Explain. (L2, CO1, 5M)

Explanation – 3M
Any two uses / advantages – 2M

2(b) Compare and contrast the Iterative Waterfall Model and the Incremental Development Model. (L2, CO1, 5M)

Explanation of both models – 3M
Any two differences – 2M

3(a) Describe the types of software development projects with suitable examples. (L2, CO1, 5M)

Explanation of project types – 3M
Any two examples – 2M

**3(b) Illustrate the working of the Rapid Application Development (RAD) model.

What types of projects are most suited for RAD, and why? (L2, CO1, 5M)
RAD model explanation – 3M
Suitable project types (any two reasons) – 2M

UNIT – II

4(a) Write short notes on Requirement Specification and Requirement Validation. (L2, CO1, 5M)

Requirement Specification – 3M
Requirement Validation – 2M

**4(b) Compare and contrast Lines of Code (LOC) and Function Point (FP) metrics.

In what scenarios would you prefer one metric over the other?** (L4, CO4, 5M)
LOC & FP explanation – 3M
Any two comparisons / preference reasons – 2M

5(a) Requirements analysis is unquestionably the most communication-intensive step in the software engineering process.Why does the communication path frequently break down? (L3, CO2, 5M)

Explanation – 3M

Reasons -2M

5b)Explain the Basic COCOMO Model. How does it estimate effort and cost for software projects?Support your answer with examples.(L4, CO4, 5M)**

Explanation – 4M

Examples -1M

UNIT – III

6(a) Compare and contrast Function-Oriented Design and Object-Oriented Design. (L4, CO4, 5M)

Explanation of both approaches – 3M
Any two comparisons – 2M

**6(b) Describe the Extreme Programming (XP) process.What are the key practices that differentiate XP? From other agile methodologies? (L2, CO1, 5M)

XP process explanation – 3M
Any two XP practices – 2M

7(a) Explain the Scrum framework in detail.Describe the roles, artifacts, and workflow involved in a typical Scrum project. (L2, CO1, 5M)
Scrum framework explanation – 3M
Any two (roles / artifacts / workflow) – 2M

**7(b) Discuss the characteristics of a Good User Interface (UI).Why is a well-designed UI important? (L3, CO3, 5M)
Any three characteristics – 4M
Why-1M

UNIT – IV

8(a) Describe black-box testing techniques: equivalence class partitioning and boundary value analysis. (L2, CO1, 5M)
Equivalence Class Partitioning – 3M
Boundary Value Analysis – 2M

**8(b) What is ISO 9000 certification? How does ISO 9000 apply to software industry,and how does it compare with the SEI CMM. ? (L3, CO3, 5M)
ISO 9000 explanation – 2M
Any two ISO vs CMM differences – 3M

9(a) Describe various software quality models.Differentiate between product metrics and process metrics. . (L2, CO1, 5M)
Software quality models explanation – 3M
Any two product vs process metric differences – 2M

**9(b) Explain the role and importance of software documentation.Distinguish between internal and external documentation. . (L2, CO1, 5M)
Documentation role explanation – 3M
Internal vs external docs (any two points) – 2M

UNIT – V

10(a) Explain the characteristics of CASE tools and their significance. (L2, CO1, 5M)

Any three characteristics – 3M
Any two significance points – 2M

**10(b) Elaborate on software maintenance process models.How do these models help estimate cost and effort? (L4, CO4, 5M)
Maintenance model explanation – 3M
Cost/effort estimation (any two points) – 2M

Code: 23DS3503

PVP 23

11a)Describe the architecture of a CASE environment.What are the key components and how do they interact to support software development and maintenance?
(L2, CO1, 5M)

Architecture explanation – 3M
Components – 2M

11(b) Explain the concept of software reverse engineering in detail. **(L2, CO1, 5M)

Reverse engineering explanation – 5M

PART-A

1.a) Define Software Crisis.
The software crisis refers to the difficulties faced in the 1960s–70s when projects were often over budget, late,unreliable, and hard to maintain due to lack of proper methods and tools.

1.b) List two notable changes in software development practices in recent years.

1. Shift from Waterfall to Agile methodologies.
2. Error Handling: Prevention vs Correction
3. Coding is Not Everything
4. Requirement specification is crucial and prioritized
5. Design Phase is Important
6. Regular Reviews
7. Systematic Testing
8. Better Documentation
9. Proper Project Planning
10. Use of Metrics
11. Adoption of DevOps and Continuous Integration/Deployment (CI/CD).

1.c) What is the requirement of Engineering Process?

The engineering process is required to ensure systematic, disciplined, and measurable development, producing high-quality software that meets customer needs within time and budget.

1.d) What are the key responsibilities of a software project manager?

1. Preparing and monitoring the project schedule.
2. Managing risks and ensuring project quality.
3. Planning and scheduling the project
4. Resource allocation, monitoring, and risk management
5. Ensuring quality and timely delivery

1.e) Define cohesion and coupling in software design.

- Cohesion: The degree to which the elements of a module belong together. *High cohesion is desirable.*
- Coupling: The degree of interdependence between modules. *Low coupling is desirable.*

1.f) State any two characteristics of a good user interface.

1. Simplicity and ease of use
2. Consistency in layout and functionality

1.g) Define the difference between internal and external software documentation.

- Internal Documentation: Comments, meaningful variable names, and code structure embedded within the source code.
- External Documentation: User manuals, design documents, test plans, and requirement specifications maintained outside the code.

1.h) Differentiate between verification and validation in software testing.

- Verification: "Are we building the product right?" – checks correctness of process (reviews, inspections).
- Validation: "Are we building the right product?" – checks correctness of final product (testing, user evaluation).

1.i) What are two major benefits of using CASE tools in software development?

1. Improves productivity and reduces development time
2. Enhances documentation accuracy and design consistency

1.j) What is component classification in software reuse?

Component classification is the process of categorizing reusable software components (e.g., by function, domain, or usage) to make them easy to identify, retrieve, and reuse in new projects.

PART B

2(a) What is the use of software development process models? Explain. (L2, CO1, 5M)

Explanation – 3M
Any two uses / advantages – 2M

A software development process model provides a structured sequence of activities required to develop software.
It acts as a framework that guides planning, organizing, executing, and controlling software projects.
Process models help define how tasks are carried out, in what order, who performs them, and how the progress is evaluated.
They ensure that development happens in a systematic, disciplined, and predictable way rather than in an ad-hoc manner.

1. Improves project planning and control

- Models provide clear stages, helping managers monitor progress and allocate resources effectively.

2. Reduces risks and development errors
   - Early identification of issues through structured phases leads to fewer defects and lower project failure rates.

2(b) Compare and contrast the Iterative Waterfall Model and the Incremental Development Model. (L2, CO1, 5M)

Explanation of both models – 3M
Any two differences – 2M

Iterative Waterfall Model:

The Iterative Waterfall Model is an improved version of the Classical Waterfall Model.

It allows going back to earlier phases to fix mistakes — which was not possible in the classical version

Incremental Development Model

Software is developed in parts (increments), not all at once.

Each increment adds new features or modules to the system.

After every increment, the software becomes more functional.

Differences:

- Delivery of software:
  - *Iterative Waterfall:* Complete system is delivered at the end.
  - *Incremental Model:* System is delivered in parts (increments).
- Customer feedback:
  - *Iterative Waterfall:* Feedback occurs after completing phases.
  - *Incremental Model:* Continuous feedback after each increment.

3(a) Describe the types of software development projects with suitable examples. (L2, CO1, 5M)

Explanation of project types – 3M
Any two examples – 2M

Software development projects generally fall into two broad categories:

1. Software Product Development:

- Generic Products: Designed for a broad, heterogeneous market aiming to satisfy common needs across many customers.

Examples include Microsoft Windos OS and Oracle Database.

These products require extensive upfront requirements analysis,substantial investment, and must support numerous configurations.

- Domain-Specific Products: Target specific industries or sectors with specialized functionalities. Examples include BANCS(banking software by TCS) and FINACLE (banking product byInfosys). These products emphasize domain expertise and adaptability within their vertical markets.

2. Software Service Projects:

- These projects focus on creating customized software to fulfill specific client requirements, adapting existing software or developing new features. They cover maintenance, enhancement, testing, consulting, or full project outsourcing.

- Examples include custom ERP solutions for enterprises, payroll software for educational institutions, or outsourced module development for telecom services.

**3(b) Illustrate the working of the Rapid Application Development (RAD) model.What types of projects are most suited for RAD, and why? (L2, CO1, 5M)
RAD model explanation – 3M
Suitable project types (ANY TWO) – 2M

RAD combines the features of: Prototyping Model (developing a working model quickly),and Evolutionary Model (incrementally improving the softwarewith feedback).

working of the Rapid Application Development (RAD)

- Development is done in short cycles (called iterations or time-boxes).
- In each cycle, a small prototype of a specific feature is built quickly.
- The customer reviews this prototype and gives feedback.
- The prototype is then refined and improved.
- The process repeats for the next feature.
- The final software is built step by step, based on continuous feedback.
- This explains the idea that prototype → feedback →build increment happens feature by feature.

Suitable project types

- Customized Software
- Non-Critical Applications
- Projects with Tight Deadlines
- Large Software with Many Modules

4(a) Write short notes on Requirement Specification and Requirement Validation. (5M) (L2, CO1, 5M)
Requirement Specification – 3 Marks
Requirement Validation – 2 Marks

Requirement Specification is the process of documenting all user and system requirements in a clear, complete, and unambiguous form.
The output of this process is the Software Requirements Specification (SRS) document, which acts as an agreement between stakeholders and developers.

It includes functional requirements, non-functional requirements, constraints, interfaces, and acceptance criteria, ensuring everyone understands what the system must do.

Requirement Validation ensures that the documented requirements are correct, complete, consistent, and feasible.
It checks whether the requirements truly represent what the customer needs.
Common validation techniques include reviews, walkthroughs, prototyping, and checklists.

*4(b) Compare and contrast Lines of Code (LOC) and Function Point (FP) metrics.

In what scenarios would you prefer one metric over the other? (L4,CO4,5)**

LOC & FP Explanation – 3 Marks

Any Two Comparisons / Preference Reasons – 2 Marks

Lines of Code (LOC):
A size metric that measures the number of physical lines of code written in a program.
It depends on the programming language and coding style.
Useful in projects where design and coding are well defined.

Function Point (FP):
A language-independent metric that measures the functional size of software based on user inputs, outputs, files, interfaces, and inquiries.
It focuses on what the software does, not how it is coded.
Useful early in the SDLC when code is not available.

Language Dependence:

LOC: Language dependent

FP: Language independent
→ Prefer FP for multi-language projects.

Stage of Estimation:

LOC: Can be used only after design/coding details are known

FP: Can be used early during requirement analysis
→ Prefer FP in early estimation stages.

5a)Requirements analysis is unquestionably the most communication-intensive step in software engineering. Why does the communication path frequently break down? (L3, CO2, 5M)*
Explanation – 3M
Reasons -2M
Requirements analysis involves continuous interaction between customers, users, analysts, and developers.
Since each group has different viewpoints, terminology, expectations, and levels of technical understanding, the communication process becomes complex.
Any misunderstanding at this stage directly affects the accuracy of requirements, leading to breakdowns in the communication path.

Reasons for communication breakdown

1. Ambiguous or unclear requirements – Stakeholders may express needs in vague terms, causing misinterpretation.
2. Different perspectives and domain knowledge gaps – Developers may not fully understand user needs, and users may not understand technical limitations.

5(b) Explain the Basic COCOMO Model. How does it estimate effort and cost for software projects? (L4,CO4,5)

Explanation – 4M

Examples -1M

The Basic COCOMO Model is a simple way to estimate how much effort, time, and cost a software project will need.
It was developed by Barry Boehm.

What it does?

It predicts effort (person-months),It predicts development time (months),It predicts cost

Based only on the size of the project measured in KLOC (thousand lines of code).

How it works (in simple steps)

Step 1: Identify project type

There are 3 types:

Organic – small and simple projects

Semi-detached – medium complexity

Embedded – complex, real-time systems

Step 2: Measure size in KLOC

Example: 20 KLOC or 50 KLOC.

Step 3: Apply formula

Basic COCOMO uses this formula:Effort (person-months):

$E=a\times(KLOC)b$

Development Time (months):

$D=c\times(E)d$

Values of a, b, c, d change based on project type.

Simple Example

Suppose a project is organic and has 50 KLOC.

Effort

$E=2.4\times(50)1.05\approx142$ person-months

Development Time

$D=2.5\times(142)0.38\approx26$ months

Cost

If one person-month costs ₹1,00,000:

Cost=142×1,00,000=₹1.42 crore

6(a) Compare and contrast Function-Oriented Design and Object-Oriented Design. (L4, CO4, 5M)

Any five comparisons – 5M

Aspect	Function-Oriented Design (FOD)	Object-Oriented Design (OOD)
Approach	Decomposes system into functions or procedures	Decomposes system into objects (entities)
Abstraction	Achieved by breaking down system into smaller functions	Achieved by grouping similar attributes and behaviors into classes (objects)
Data Handling	Centralized, shared global data structures	Data is decentralized and encapsulated within objects
Execution	Structured analysis and design using data flow diagrams	Carried out using UML diagrams
Design Direction	Top-down approach	Bottom-up approach
Starting Point	Begins from use case diagrams and scenarios	Begins by identifying objects and classes
Decomposition Level	Decompose at function/procedure level	Decompose at class level
Modularity	Generally less modular due to centralized data	Generally highly modular with encapsulation
Reusability	Generally less reusable	Supports high reusability via encapsulation and inheritance
Coupling	Functions tend to be more tightly coupled	Objects tend to be loosely coupled

Inheritance	Not applicable	Supported, allowing inheritance of attributes and behaviors
Suitable For	Computation-sensitive applications	Evolving systems mimicking real-world business domains

**6(b) Describe the Extreme Programming (XP) process.What are the key practices that differentiate XP? From other agile methodologies? (L2, CO1, 5M)

XP process explanation – 3M

Any two XP practices – 2M

Developed by Kent Beck (1990s).

Emphasizes teamwork, communication, feedback.

- Suitable for small to medium projects with frequent changes.

Key Practices

- Small releases

-Simple design

- Pair programming

- Test-driven development (TDD)

- Refactoring

- Collective code ownership

- Continuous integration

- On-site customer

- Coding standards

7(a) Explain the Scrum framework in detail.Describe the roles, artifacts, and workflow involved in a typical Scrum project. (L2, CO1, 5M)

Scrum framework explanation – 3M

Any two (roles / artifacts / workflow) – 2M

Scrum

Iterative and incremental Agile framework.

Focus on project management (not technical coding).

- Handles rapidly changing requirements.

Scrum roles

Product Owner: Maintains product backlog, priorities

Scrum Master: Ensures Scrum practices, removes blockers.

- Development Team: Cross-functional, delivers increments.

Scrum Artifacts

- Product Backlog: Ordered list of requirements.

Sprint Backlog: Items chosen for sprint.

- Increment: Working product at sprint end.

7(b) Discuss the characteristics of a Good User Interface (UI).Why is a well-designed UI important? (L3, CO3, 5M)

Code: 23DS3503
Any three characteristics – 4M
Why-----1M

Characteristics of a Good User Interface Design:

1. Clarity: Information and options must be clearly visible and understandable to reduce confusion.

2. Consistency: The interface should maintain uniform design and behavior across all screens, enhancing learnability.

3. Responsiveness: The UI must respond promptly to user actions, essential for real-time performance.

4. User Control: Users should have control over interactions with options to undo or redo actions.

5. Simplicity: Avoid unnecessary complexity; provide essential features prominently to facilitate task completion.

6. Feedback: Immediate and informative feedback guides users on system status and result of actions.

7. Accessibility: Supports use by people with varying abilities, including those with disabilities.

Why::It enhances usability, boosts user satisfaction, and helps retain customers. A well-designed interface improves navigation, fosters brand identity, and ensures accessibility

8(a) Describe black-box testing techniques: equivalence class partitioning and boundary value analysis. (L2, CO1, 5M)
Equivalence Class Partitioning – 3M
Boundary Value Analysis – 2M

Also known as functional testing or specification-based testing. The tester does not need to know the internal structure of the program. Only the specifications (requirements, possible valid/invalid inputs, expected outputs) are used.

Used mainly for validation.

Techniques

A)Equivalence Class Partitioning (ECP)

B) Boundary Value Analysis (BVA)

Equivalence Class Partitioning (ECP)

Divide input into groups (classes) where system behaves similarly.

Test only one value from each class to save effort.

Example:
Suppose input is student marks (0–100):

Valid Class: 0–100

Invalid Class: <0 and >100
Test with one value from each: e.g., 50 (valid), -5 (invalid), 150 (invalid)

Boundary Value Analysis (BVA)

Many errors occur at the edges of input range.

Test with values at, just below, and just above boundaries.

Example:
Marks range: 0–100
Test values: -1, 0, 1 (lower boundary) and 99, 100, 101 (upper boundary)

**8(b) What is ISO 9000 certification? How does ISO 9000 apply to software industry,and how does it compare with the SEI CMM. ? (L3, CO3, 5M)

Code: 23DS3503

ISO 9000 explanation – 2M
Any two ISO vs CMM differences – 3M

What is ISO 9000 Certification?

ISO 9000 is a set of international standards for quality management systems.

It helps organizations ensure they consistently meet customer and regulatory requirements.

Getting ISO 9000 certified means your company has a well-documented and well-managed process for delivering quality products or services.

ISO 9000 in the Software Industry

In software, ISO 9000 ensures that:

Requirements are clearly defined.

Development and testing follow documented procedures.

Customer feedback is recorded and acted upon.

It doesn't tell you how to develop software, but it makes sure your process is organized and quality-focused.

Feature	ISO 9000	SEI CMM
Focus	Quality management system	Software process improvement
Scope	All industries	Software industry
Certification	Yes, by external auditors	No formal certification; it's a model
Approach	Customer-focused, documentation-heavy	Process maturity and continuous improvement
Levels	No levels	5 maturity levels
Who uses it?	Any organization	Software development companies

9(a) Describe various software quality models.Differentiate between product metrics and process metrics. . (L2, CO1, 5M)

Software quality models explanation – 3M
Any two product vs process metric differences – 2M

Software Quality Models help us understand what makes software "good" or "bad."

Some commonly used models are:

1. McCall's Quality Model
   Classifies quality into three categories: Product Operation (correctness, reliability), Product Revision (maintainability), and Product Transition (portability).
2. Boehm's Quality Model
   Emphasizes high-level characteristics like utility, portability, reliability, and maintainability. It refines quality through hierarchical attributes.
3. ISO 9126 Quality Model (Now ISO/IEC 25010)
   Defines six major quality characteristics: Functionality, Reliability, Usability, Efficiency, Maintainability, Portability.

Product Metrics vs Process Metrics

Product Metrics	Process Metrics
Measure the quality of the final software product (e.g., size, defects, complexity).	Measure the quality of the software development process (e.g., effort, schedule variance, defect arrival rate).

Product Metrics	Process Metrics
Used to evaluate what has been built.	Used to evaluate how the software was built.

.

9(b) Explain the role and importance of software documentation.Distinguish between internal and external documentation. . (L2, CO1, 5M)
Documentation role explanation – 3M
Internal vs external docs (any two points) – 2M
Role and Importance of Software Documentation (3M)

1. Helps understand the software – Explains requirements, design, and code, so developers and new team members can easily understand the system.
2. Helps in maintenance – Makes it easy to fix bugs, update features, and modify the software in the future.
3. Improves communication – Gives clear information to developers, testers, users, and managers, reducing confusion and errors.

Internal Documentation	External Documentation
Written *inside the code* (comments, clear variable names).	Written *outside the code* (SRS, design docs, user manuals).
Used mainly by developers.	Used by developers, testers, users.

10(a) Explain the characteristics of CASE tools and their significance. (L2, CO1, 5M)

Any three characteristics – 3M
Any two significance points – 2M

CASE Environment – an integrated workspace connecting multiple CASE tools.

In this environment:All design files, code, and documents are stored in one big central repository

The tools are connected — so when a designer changes something, the coder and tester instantly see it!

• Central Repository – stores all design and project data for shared access.

Benefits:
- Cost and time savings (30–40% effort reduction).
- Improved quality and documentation.
- Reduced errors and rework.
- Enhanced teamwork and easier maintenance.

• Example: IBM Rational Suite integrates design, coding,and testing tools.

**10(b) Elaborate on software maintenance process models.How do these models help estimate cost and effort? (L4, CO4, 5M)
Maintenance model explanation – 3M
Cost/effort estimation (any two points) – 2M
Different approaches used to manage maintenance work:
1. Quick Fix Model – fixes urgent bugs immediately.
2. Iterative Enhancement Model – planned improvements in cycles.
3. Reuse-Oriented Model – updates using reusable components.
How These Models Help Estimate Cost & Effort?

1. Clear Steps for Planning:
   Each model has defined phases—analysis, modification, testing—so the amount of work can be easily estimated.
2. Predicts the Type of Maintenance Work:
   Models show whether the task is a small fix, a major enhancement, or a full redesign. This helps estimate the required time, resources, and cost.
3. Helps Allocate Resources Effectively (optional):
   Knowing the model helps decide how many people and skills are needed, making cost estimation more accurate.

11a)Describe the architecture of a CASE environment. What are the key components and how do they interact to support software development and maintenance?
(L2, CO1, 5M)

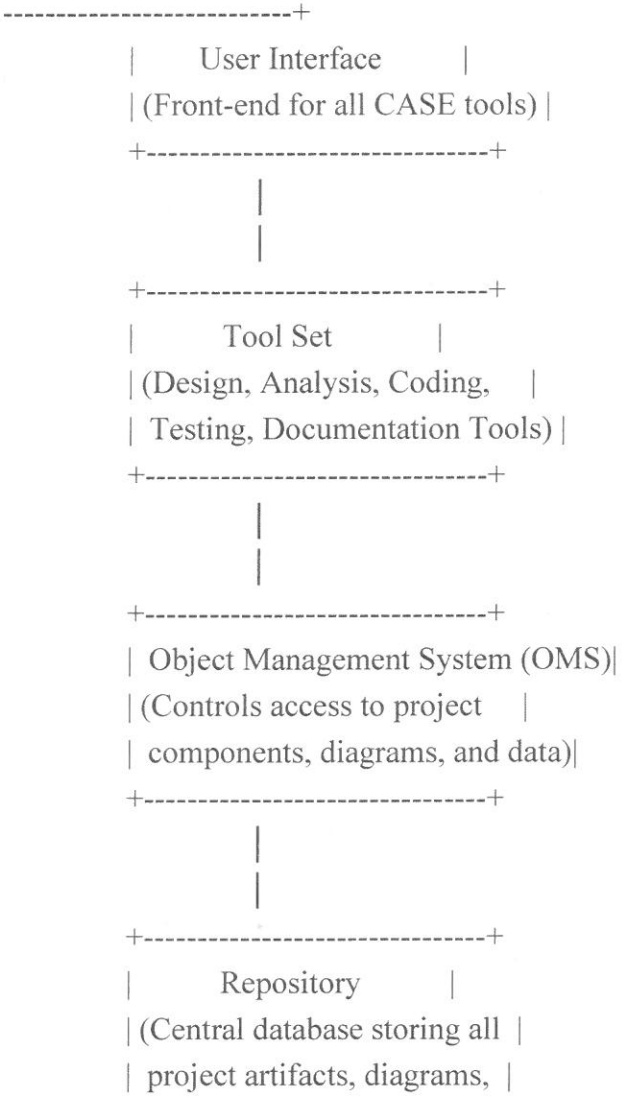
Architecture explanation – 3M
Components – 2M

• Main Components:

1. User Interface – consistent access to all tools (IBM Rational UI).
2. Tool Set – includes design, coding, and testing tools (StarUML, Selenium).
3. Object Management System (OMS) – manages project entities and code.
4. Repository – central storage for all project information.

• Example: Repository stores UML diagrams and source code for collaboration.

Architecture::



- | code, and documentation) |
- +----------+

11(b) Explain the concept of software reverse engineering in detail. (L2, CO1, 5M)

Reverse engineering explanation – 5M

- Process of analyzing existing software to understand its design and structure.
- Used when design documents are missing or outdated.
- Steps: Code analysis → Design recovery → Documentation creation.
- Helps in maintaining or modernizing old software systems.
- Example: Converting old COBOL code into modern UML design.